An Unstructured-Grid, Parallel, Projection Solver for Computing Low-Speed Flows

Mark A. Christon and Daniel E. Carroll
Computational Physics R&D Department
Sandia National Laboratories
M/S 0819, P.O. Box 5800
Albuquerque, New Mexico 87185-0819

Summary

This paper presents an overview of the issues associated with applying a domain-decomposition message-passing paradigm to the parallel implementation of both explicit and semi-implicit projection algorithms. The use of an element-based domain decomposition with an efficient solution strategy for the pressure field is shown to yield a scalable, parallel solution method capable of treating complex flow problems where high-resolution grids are required. In addition, the use of an SSOR or Jacobi preconditioned conjugate gradient solver with an A-conjugate projection reduces the computational time for the solution of the pressure field, and yields parallel efficiencies above 80% for computations with $O(250)$ elements per processor. The parallel projection solver is verified using a series of 2-D and 3-D benchmarks designed to evaluate time-accurate flow solution methods. Finally, the extension of the projection algorithm to reacting flows is demonstrated for a time-dependent vortex-shedding problem.

Introduction

The incompressible Navier-Stokes equations pose significant challenges for the design and implementation of efficient parallel flow solution algorithms due, in part, to the global elliptic nature of the pressure imposed by the incompressibility constraint, $\nabla \cdot \mathbf{u} = 0$. In addition to the algorithmic difficulties, time dependent flows in complex geometry can require meshes with over $O(10^6)$ grid points even for moderate Reynolds numbers. The concomitant demands for spatial and temporal resolution in large eddy simulations further drives the need for efficient, scalable flow solvers. Although fully-coupled solution strategies are available, the cost of such methods is prohibitive for time-dependent simulations where high-resolution grids are required. Projection methods offer a computationally tractable alternative to fully-coupled flow solution algorithms. The primary focus of this work is upon the application of a domain-decomposition message-passing (DDMP) approach to solving the transient, viscous, incompressible, Navier-Stokes equations using projection methods in a finite element framework. The algorithm-to-architecture mapping issues for a parallel, second-order, projection algorithm for transient, incompressible flow simulations are outlined in the context of both explicit and semi-implicit time integrators. A generalized finite element assembly paradigm provides the framework for implementation of the parallel flow solver and the concomitant linear solvers. Sample computations are presented with scaled efficiency curves to illustrate the scalability of the parallel flow solver.

Formulation

The goal for projection algorithms is to decouple the pressure and velocity fields in as legitimate a manner as possible in the hope of providing an efficient computational method for high-resolution, transient, incompressible flow simulations. In practice, the action of the projection is to remove the portion of the approximate velocity field that is not divergence free. In effect, the projection is achieved by separating the velocity field into div-free, $\mathbf{u}$, and curl-free, $\nabla\lambda$, components using a Helmholtz

decomposition, i.e., $\tilde{\mathbf{u}} = \mathbf{u} + \nabla\lambda$ where $\tilde{\mathbf{u}}$ is a non-solenoidal velocity field. Here, the *optimal* Projection-2 (P2) method identified by Gresho [2] forms the basis for the parallel projection algorithms.

The spatial discretization of the conservation equations is achieved using a stabilized Q_1P_0 element. The methods for obtaining the basic weak-form of the conservation equations are well known and will not be repeated here (see for example, Hughes [6]). The spatially discrete form of the incompressible mass and momentum conservation equations are

$$C^T\mathbf{u} = 0, \tag{1}$$

$$M\dot{\mathbf{u}} + A(\mathbf{u})\mathbf{u} + K\mathbf{u} + CP = \mathbf{F}, \tag{2}$$

where $\mathbf{u} = (u, v, w)$ is the velocity, p is the pressure, ρ is the mass density, and $P = p/\rho$. Here M is the unit mass matrix, $A(\mathbf{u})$ and K are the advection and the viscous diffusion operators respectively, and $\mathbf{F}$ is the body force. C is the gradient operator, and C^T is the divergence operator.

The P2 algorithm proceeds as follows. Given a div-free velocity, $\mathbf{u}^n$, and its corresponding pressure field, P^n, solve the momentum equations for an approximate velocity field at time level $n+1$.

$$[M + \Delta t\theta K]\tilde{\mathbf{u}}^{n+1} = [M - \Delta t(1-\theta)K]\mathbf{u}^n + \Delta t\{\theta\mathbf{F}^{n+1} + (1-\theta)\mathbf{F}^n - A\mathbf{u}^n - MM_L^{-1}CP^n\} \tag{3}$$

The discrete statement of the Helmholtz velocity decomposition with the concomitant div-free constraint, $C^T\mathbf{u}^{n+1} = 0$, is

$$\begin{bmatrix} M_L & C \\ C^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{u}^{n+1} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} M_L\tilde{\mathbf{u}}^{n+1} \\ 0 \end{Bmatrix}. \tag{4}$$

Here, M_L is the row-sum lumped, i.e., diagonalized, mass matrix, and includes the prescription of essential velocity boundary conditions. In Eq. (4), $\lambda = \Delta t\{\theta P^{n+1} - (1-\theta)P^n\}$ where $0 \le \theta \le 1$ determines the order of accuracy of the time integrator. Typically $\theta = 1/2$ for second-order temporal accuracy. The Schur complement of the saddle-point problem, a consistent pressure Poisson equation (PPE), may also be formed explicitly,

$$[C^TM_L^{-1}C]\lambda = C^T\tilde{\mathbf{u}}^{n+1}. \tag{5}$$

Given the approximate velocity, $\tilde{\mathbf{u}}^{n+1}$, either Eq. (4) may be solved for both $\mathbf{u}^{n+1}$ and λ, or Eq. (5) may be solved to yield λ. If Eq. (5) has been solved, then an orthogonal projection of the discrete, approximate velocity onto a discretely div-free subspace is performed, i.e., $\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - M_L^{-1}C\lambda$. Computational experiments with a gradient-based saddle-point solver has shown that the computational cost associated with an element-by-element matrix-free solution strategy for Eq. (5) and the saddle point solver are equivalent. After the velocity update, an updated pressure at time level $n+1$ is obtained via $P^{n+1} = P^n + 2\lambda/\Delta t$.

There are several modifications to the basic finite element formulation that derive from the treatment of the advective terms. First, an advective flux-limiting procedure is employed to preserve monotonicity. For advective flows, it is well known that the use of a backward-Euler treatment of the advective terms introduces excessive diffusion. Similarly, Gresho et al.[4] have shown that a forward-Euler treatment of the advective terms results in negative diffusivity, or an under-diffusive scheme. In order to remedy this problem, balancing-tensor diffusivity (BTD), which is derived from a Taylor series analysis to exactly balance the diffusivity deficit and to deliver second-order accuracy in time, is adopted.

The time step for the explicit projection algorithm is restricted to a combined advective-diffusive limit [4]. However the semi-implicit projection algorithm must only respect a relaxed convective stability limit where $CFL \leq O(5-10)$ as described in Gresho and Chan [3]. Because of the semi-implicit treatment of diffusion, there is no diffusive stability constraint. Computational experiments have demonstrated that $CFL \leq 5$ is a reasonable tradeoff between accuracy and computational cost.

Algorithm Mapping

Domain-decomposition is the process of sub-dividing the spatial domain into sub-domains that can be assigned to individual processors for the subsequent parallel solution process. In general, the primary requirements for mesh decomposition are that the sub-domains be defined in such a way that the computational load is uniformly distributed across the available processors, and that the inter-processor communication is minimized. In this work, the decomposition of the finite element mesh into sub-domains is accomplished using the multilevel graph partitioning tools available in CHACO [5].

In both the explicit and semi-implicit projection algorithms, the solution of the saddle-point problem, Eq. (4), or its Schur complement, Eq. (5), can consume $80-90\%$ of the cpu time per time step. In order to exploit the finite element assembly process (see [6]) for parallelization while load-balancing the solution procedure, the dual graph of the finite element mesh, i.e., the connectivity of the dual grid is used to perform a non-overlapping element-based domain decomposition. Implicit in this choice of a domain decomposition strategy is the idea that elements are uniquely assigned to processors while the nodes at the sub-domain interfaces are stored redundantly in multiple processors. Figure 1a illustrates a non-overlapping, element-based decomposition for a two-dimensional vortex-shedding mesh and an airfoil mesh, each load-balanced for sixteen processors.

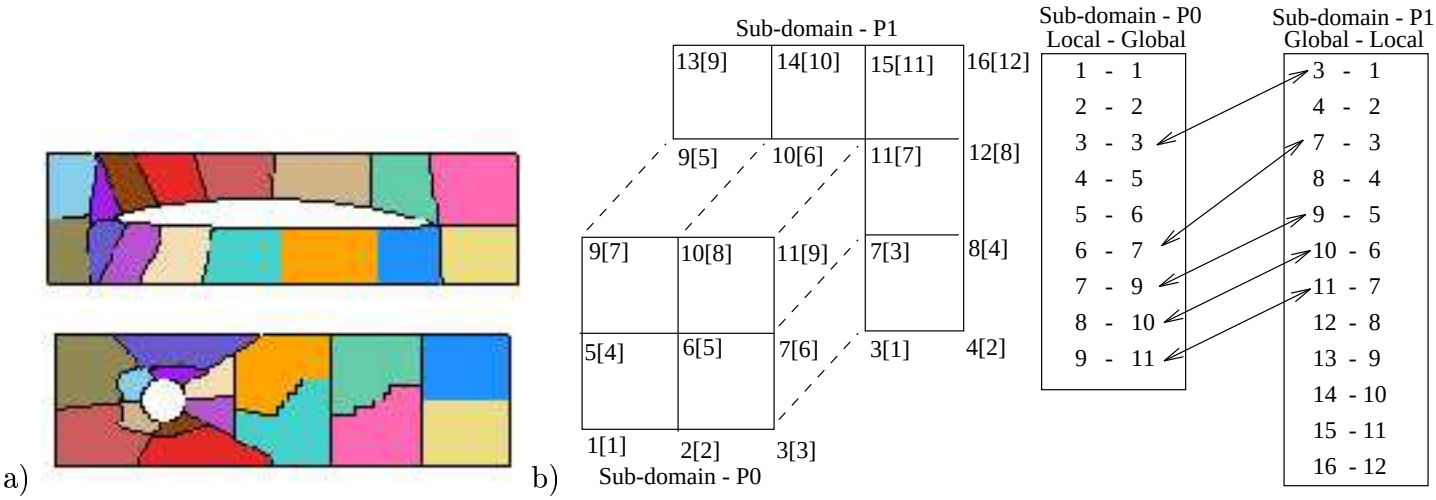

Figure 1: a) Sixteen processor spatial domain decomposition for a cylinder vortex-shedding mesh and an airfoil mesh, and b) a two processor example sub-domain mapping and parallel assembly.

The finite element assembly procedure is an integral part of any finite element code and consists of a nodal gather operation, an add operation, and a subsequent scatter back to the global memory locations. The assembly procedure is used to both form global coefficient matrices, right-hand-side vectors, and matrix-vector products in an element-by-element sense. In the case of the explicit time integration algorithm, the emphasis is upon the assembly of the element level contributions to the global right-hand-side vector,

$$\hat{\mathbf{F}} = \mathcal{A}_{p=0}^{Np-1}\{\mathcal{A}_{e=1}^{Nel^p}\{\mathbf{f}_e^n - K_e\mathbf{u}_e^n - A_e(\bar{\mathbf{u}})\mathbf{u}_e^n\}\}. \tag{6}$$

Here $\mathcal{A}$ is the finite element assembly operator, Nel^p is the number of elements assigned to each processor, p, and Np is the number of processors.

The parallel assembly procedure may be viewed as a generalized form of the finite element assembly algorithm that requires inter-processor communication. As an example, consider the distributed mesh and the assignment of the global nodes to two processors as shown in Figure 1b. Here, the on-processor local node numbers are enclosed in brackets to the right of the global node number (global node 10 in sub-domain $P0$ is local node [8]). The parallel assembly of the sub-domain boundary nodes is achieved with the use of the local-to-global mapping required for the gather-add-scatter assembly procedure. The arrows between global node numbers identify a send-receive message pair.

Direct solution methods have been shown to be effective for solving the PPE because a single off-line factorization may be used with one resolve per time step. However, direct solvers lack scalability with respect to problem size due to memory requirements and the operations count – both a function of the bandwidth of the PPE. As an alternative, four conjugate gradient based algorithms have been implemented with an outer A-conjugate projection. The four solvers are an element-by-element Jacobi preconditioned conjugate gradient matrix-free method (EBE-JPCG), a Jacobi preconditioned conjugate gradient method (JPCG), a symmetric successive over-relaxation preconditioned conjugate gradient method (SSOR-PCG), and a symmetric over-relaxation preconditioned conjugate gradient method using the Eisenstat transformation [1] (ESSOR-PCG). The iteration count and normalized grind time for 2-D and 3-D are shown in Table 1 where the number of projection vectors ranges from 0 to 50.

	EBE-JPCG		JPCG		SSOR-PCG		ESSOR-PCG	
No. of Vectors	N_{IT}	Grind Time	N_{IT}	Grind Time	N_{IT}	Grind Time	N_{IT}	Grind Time
0	189	5.22	189	4.32	69	3.64	69	2.77
10	71	2.60	71	2.12	25	1.86	25	1.61
25	39	1.89	39	1.62	14	1.47	14	1.34
50	25	1.64	25	1.45	10	1.39	10	1.29

a) 7040 elements, 7224 nodes, and 1/2 bandwidth = 161

	EBE-JPCG		JPCG		SSOR-PCG		ESSOR-PCG	
No. of Vectors	N_{IT}	Grind Time	N_{IT}	Grind Time	N_{IT}	Grind Time	N_{IT}	Grind Time
0	52	1.58	53	1.33	29	1.44	22	1.20
10	30	1.28	30	1.15	15	1.18	12	1.06
25	30	1.28	30	1.15	14	1.15	11	1.05
50	26	1.23	26	1.12	12	1.12	10	1.04

b) 7840 elements, 9120 nodes, and 1/2 bandwidth = 576

Table 1: Effect of the A-Conjugate projection on the PPE solution time for a) 2-D vortex shedding, and b) a 3-D juncture flow. N_{IT} is the average number of iterations per time step, and "Grind Time" is the cpu time per time step per element. The grind time has been normalized with respect to the grind time using a direct solver[1] with one resolve per time step. All timing are based upon calculations using 1000 time steps.

[1]The parallel-vector solver (PVS) [7] was used to normalize the grind times for the A-conjugate projection.

Results

Scaled efficiency results for a series of $Re = 800$, 2-D backward facing step and $Re = 100$, 3-D juncture flows are shown in Figure 2. In each calculation, approximately 250 elements were assigned to each processor. The efficiency for each computation is computed based upon the percentage of measured time spent performing communication versus computation. In these scaling experiments, the EBE-JPCG and SSOR-PCG pressure solvers each used 0 and 10 projection vectors. The effect of the sub-domain SSOR preconditioning and the A-conjugate projection was to improve the parallel efficiency due to increased computational work on each processor and a reduced CG iteration count. The general effect of the A-conjugate projection was to reduce the average number of iterations required to solve the PPE problem resulting in better overall parallel efficiency.

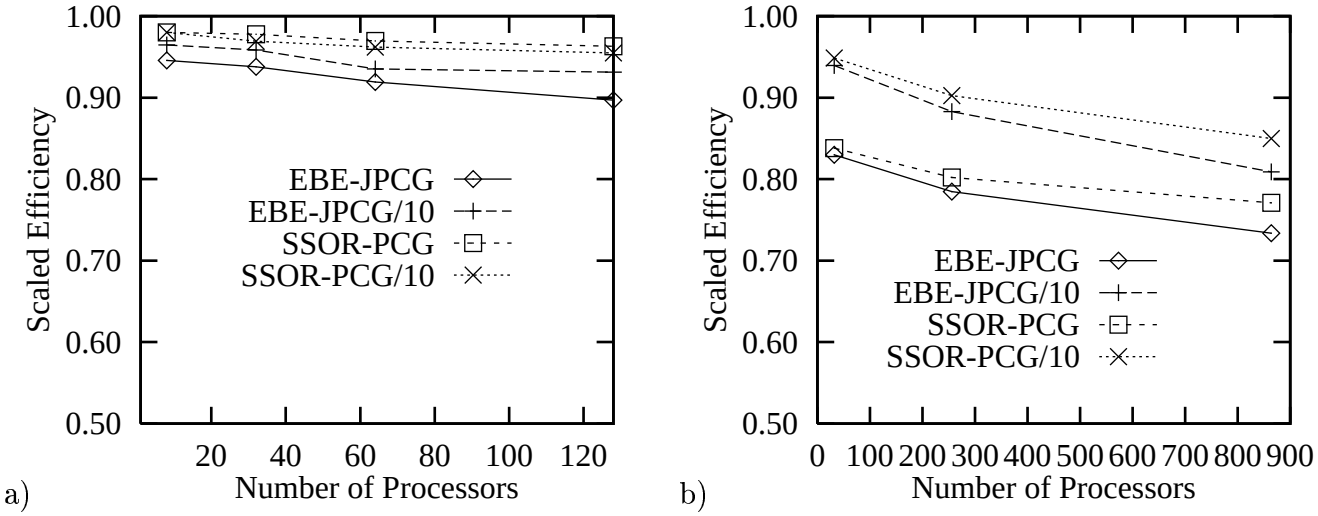

Figure 2: Scaled efficiency results for a) $Re = 800$ 2-D backward facing step, and b) $Re = 100$ 3-D post and plate juncture.

Figure 3 below shows snapshots of several computations that were motivated by a suite of benchmark problems used as "round-robin" tests to evaluate incompressible flow solution methods developed under the Deutsche Forschungsgemeinschaft (DFG) Priority Research Program, "Flow Simulation on High Performance Computers". The benchmark problems considered here were designed specifically to test time-accurate solution methods. Figure 3a shows a snapshot of the 2-D vorticity field for a $Re = 100$ flow in a channel with a circular cylinder. The Strouhal number for the 2-D computation is 0.293 while the experimentally determined value is 0.287 ± 0.003. In contrast to the 2-D results, the vorticity snapshot for the 3-D calculation in Figure 3b shows considerable mixing in the downstream wake. The snapshot of the helicity field $(\mathbf{u} \cdot \omega)$ in Figure 3d shows the presence of longitudinal vortices formed at the cylinder–wall juncture. Snapshots of the temperature and fuel mass fraction for the chemically reacting vortex-shedding problem are shown in Figure 3c.

Conclusions

This effort has demonstrated the effectiveness of the domain-decomposition message-passing paradigm for the semi-implicit projection algorithm for the incompressible, viscous Navier-Stokes equations. The decomposition based upon elements rather than nodes yields good parallel efficiencies due to the high proportion of computational work associated with solving the pressure Poisson equation. Reasonable parallel efficiencies are achieved with this approach even for fine-grained parallel computations with $O(250)$ elements per processor.

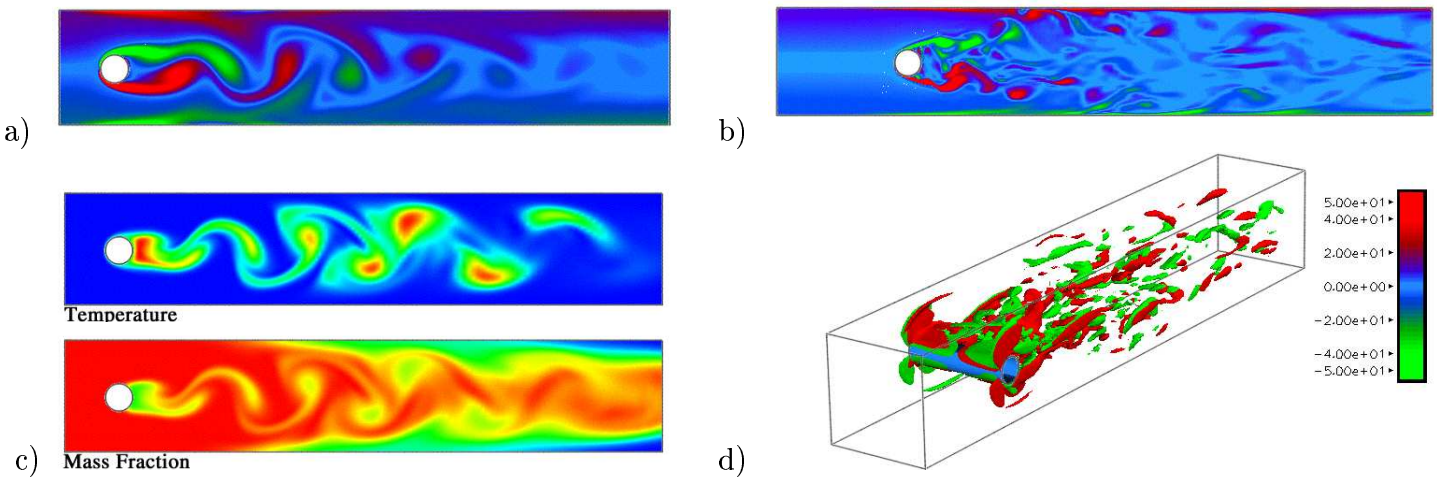

Figure 3: Results of benchmark computations showing a) 2-D vorticity field for $Re = 100$, b) z-vorticity field on a vertical plane of a 3-D, $Re = 100$ flow past a cylinder in a channel, c) temperature and fuel mass concentration for reacting, 2-D case and, d) iso-surfaces of the 3-D helicity ($\mathbf{u} \cdot \omega$) field. In the chemically reacting case, buoyancy has been neglected. (The colormap and scale applies to the vorticity and helicity fields.)

Acknowledgements

This work was supported by the U.S. DOE under Contract DE-AC04-94AL85000.

References

[1] Eisenstat, S. C. (1983), A note on the generalized conjugate gradient method, SIAM Journal of Numerical Analysis, Vol. 20, pp. 358–361.

[2] Gresho, P. M. (1990), On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 1: Theory, International Journal for Numerical Methods in Fluids, Vol. 11, pp. 587–620.

[3] Gresho, P. M., and Chan, S. T. (1990), On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 2: Implementation. International Journal for Numerical Methods in Fluids, Vol. 11, pp. 621–659.

[4] Gresho, P. M., Chan, S. T., Lee, R. L., and Upson, C. D. (1984), A modified finite element method for solving the time-dependent, incompressible Navier-Stokes equations. part 1: Theory. International Journal for Numerical Methods in Fluids, Vol. 4, pp. 557–598.

[5] Hendrickson, B., and Leland, R. (1993), The chaco user's guide - version 1.0, Tech. Rep. SAND93-2339, Sandia National Laboratories Report.

[6] Hughes, T. J. R. (1983), The Finite Element Method, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[7] Storaasli, O., Nguyen, D., and Agarwal, T. (1990), A parallel-vector algorithm for rapid structural analysis on high-performance computers, Tech. Rep. 102614, NASA.